CLAIMS

What is claimed is:

1. A transform system for graphics processing, comprising:

(a) an input buffer adapted for being coupled to a vertex attribute buffer for receiving vertex data therefrom;

(b) a multiplication logic unit having a first input coupled to an output of the input buffer;

(c) an arithmetic logic unit having a first input coupled to an output of the multiplication logic unit;

(d) a register unit having an input coupled to an output of the arithmetic logic unit;

(e) an inverse logic unit including an input coupled to the output of the arithmetic logic unit or the register unit for performing an inverse or an inverse square root operation;

(f) a conversion module coupled between an output of the inverse logic unit and a second input of the multiplication logic unit, the conversion module adapted to convert scalar vertex data to vector vertex data;

(g) memory coupled to the multiplication logic unit and the arithmetic logic unit, the memory having stored therein a plurality of constants and variables for being used when processing the vertex data; and

(h) an output converter coupled to the output of the arithmetic logic unit and adapted for being coupled to a lighting module to output the processed vertex data thereto.

2. The system as recited in claim 1, wherein the memory is coupled to the second input of the multiplication logic unit.

3. The system as recited in claim 1, wherein the input of the inverse logic unit is coupled to the output of the arithmetic logic unit.

4. The system as recited in claim 1, wherein the inputs of the multiplication logic unit include multiplexers.

5. The system as recited in claim 1, wherein at least one of the inputs of the arithmetic logic unit includes a multiplexer.

6. The system as recited in claim 1, wherein the memory has a write terminal coupled to the output of the arithmetic logic unit.

7. The system as recited in claim 1, wherein the output of the multiplication logic unit has a feedback loop coupled to the first input thereof.

8. The system as recited in claim 1, wherein the output of the register unit is coupled to the first input of the multiplication logic unit.

9. The system as recited in claim 8, wherein the output of the register unit is coupled to the second input of the multiplication logic unit.

10. The system as recited in claim 1, wherein the output of the arithmetic logic unit has a feedback loop connected to the second input thereof.

11. The system as recited in claim 10, wherein the feedback loop has a delay coupled thereto.

12. The system as recited in claim 1, wherein the multiplication logic unit is capable of performing a rotate operation on vector vertex data.

13. The system as recited in claim 1, wherein the inverse logic unit is capable of clamping a value of an inverse operation if the value of the inverse operation meets predetermined criteria.

14. The system as recited in claim 1, wherein included are six input buffers coupled to the first input of the multiplication logic unit.

15. The system as recited in claim 1, wherein the multiplication logic unit includes four multipliers coupled in parallel.

16. The system as recited in claim 1, wherein the arithmetic logic unit includes three adders coupled in parallel and series.

17. The system as recited in claim 1, wherein the register unit includes four sets of registers each having an output coupled to a first input of an associated multiplexer which has a second input coupled to the input of the corresponding set of registers.

18. The system as recited in claim 1, wherein the register unit is threaded.

19. The system as recited in claim 1, wherein the output converter is adapted to be coupled to the lighting module via output buffers.

20. The system as recited in claim 1, wherein a register is coupled between the output of the inverse logic unit and an input of the conversion unit.

21. The system as recited in claim 20, wherein the register is threaded.

22. The system as recited in claim 1, wherein the register unit is capable of being masked at a vector component level.

23. A system for handling scalar and vector components during graphics processing, comprising:

(a) a vector operation module for receiving vertex data in the form of vectors and performing vector operations on the vector vertex data;

(b) a conversion module coupled to the vector operation module for converting scalar vertex data from the vector operation module into vector vertex data; and

(c) a register coupled to the vector operation module for storing an output of the vector operation module for feeding the output back to the vector operation module.

24. The system as recited in claim 23, wherein the vector operation module includes at least one of multiplier and an adder.

25. The system as recited in claim 23, wherein zero latency is achieved by bypassing the register.

26. The system as recited in claim 25, wherein the register includes a vector component write mask for generating vector vertex data.

27. The system as recited in claim 23, and further comprising a scalar operation module adapted for executing scalar operations on an output of the vector operation module, thereby rendering vertex data in the form of scalars.

28. The system as recited in claim 27, wherein the scalar operations include inverse or inverse square root operations.

29. A method for handling scalar and vector components during graphics processing, comprising:

(a) receiving vertex data in the form of vectors;

(b) performing vector operations on the vector vertex data;

(c) converting scalar vertex data resulting from the vector operations into vector vertex data;

(d) storing an output of the vector operations; and

(e) performing additional vector operations on the stored output of the vector operations.

30. The method as recited in claim 29, wherein the vector operations include multiplication or addition operations.

31. The method as recited in claim 29, wherein the vector operations are performed on the output of the vector operations with zero latency.

32. The method as recited in claim 31, wherein the output of the vector operations is stored in a register unit, and the zero latency is achieved by bypassing the register unit.

34. The method as recited in claim 29, and further comprising executing scalar operations on an output of the vector operations, thereby rendering vertex data in the form of scalars.

35. The method as recited in claim 34, wherein the scalar operations include inverse or inverse square root operations.

36. The method as recited in claim 34, and further comprising extracting scalar vertex data from the output of the vector operations if the output is in the form of vectors.

37. The method as recited in claim 36, wherein the extraction is carried out by a multiplexer.

38. The method as recited in claim 29, wherein the received vertex data is manipulated by a multiplexer.

39. A computer program embodied on a computer readable medium for handling scalar and vector components during graphics processing, comprising:
(a) a code segment for receiving vertex data in the form of vectors;
(b) a code segment for performing vector operations on the vector vertex data;
(c) a code segment for converting scalar vertex data resulting from the vector operations into vector vertex data;
(d) storing an output of the vector operations; and
(e) performing additional vector operations on the stored output of the vector operations.

40. The computer program as recited in claim 39, wherein the vector operations include multiplication or addition operations.

41. The computer program as recited in claim 39, wherein the vector operations are performed on the output of the vector operations with zero latency.

42. The computer program as recited in claim 41, wherein the output of the vector operations is stored in a register unit, and the zero latency is achieved by bypassing the register unit.

43. The computer program as recited in claim 39, and further comprising a code segment for executing scalar operations on an output of the vector operations, thereby rendering vertex data in the form of scalars.

44. The computer program as recited in claim 43, wherein the scalar operations include inverse or inverse square root operations.

45. The computer program as recited in claim 43, and further comprising a code segment for extracting scalar vertex data from the output of the vector operations if the output is in the form of vectors.

46. The method as recited in claim 45, wherein the extraction is carried out by a multiplexer.

47. The method as recited in claim 39, wherein the received vertex data is manipulated by a multiplexer.

48. A method for performing a blending operation during graphics processing in a hardware-implemented graphics pipeline, comprising:

(a) receiving a plurality of matrices, a plurality of weight values each corresponding with one of the matrices, and vertex data in a buffer;

(b) calculating a sum of a plurality of products with each product calculated by the multiplication of the vertex data, one of the matrices, and the weight corresponding to the matrix, wherein the calculation is executed on a single integrated circuit; and

(c) outputting the sum of products for additional processing.

49. The method as recited in claim 48, wherein the matrices include model view matrices.

50. The method as recited in claim 49, wherein the additional processing includes multiplying the sum of products by a composite matrix for displaying purposes.

51. The method as recited in claim 49, wherein the additional processing includes a lighting operation.

52. The method as recited in claim 48, wherein the matrices include inverse matrices and the vertex data includes a normal vector.

53. The method as recited in claim 48, wherein the single integrated circuit includes: a multiplication logic unit having a first input coupled to an output

of the buffer for receiving the vertex data; an arithmetic logic unit having a first input coupled to an output of the multiplication logic unit; a register unit having an input coupled to an output of the arithmetic logic unit; memory coupled to the multiplication logic unit and the arithmetic logic unit, the memory having stored therein a plurality of constants and variables for being when processing the vertex data.

54. The system as recited in claim 48, wherein the single integrated circuit includes: a multiplication logic unit having a first input coupled to an output of the buffer; an arithmetic logic unit having a first input coupled to an output of the multiplication logic unit; a register unit having an input coupled to an output of the arithmetic logic unit; an inverse logic unit including an input coupled to the output of the arithmetic logic unit or the register unit for performing an inverse or an inverse square root operation; a conversion module coupled between an output of the inverse logic unit and a second input of the multiplication logic unit, the conversion module adapted to convert scalar vertex data to vector vertex data; memory coupled the multiplication logic unit and the arithmetic logic unit, the memory having stored therein a plurality of constants and variables for being used when processing the vertex data; and an output converter coupled to the output of the arithmetic logic unit for being coupled to a lighting module to output the processed vertex data thereto.

55. A system for performing a blending operation during graphics processing in a graphics pipeline, comprising:

(a) a buffer for receiving a plurality of matrices, a plurality of weight values each corresponding with one of the matrices, and vertex data;

(b) a single integrated circuit coupled to the buffer for calculating a sum of a plurality of products with each product calculated by the multiplication of the vertex data, one of the matrices, and the weight corresponding to the matrix; and

(c) wherein the sum of products is outputted from the single integrated circuit for additional processing.

56. The system as recited in claim 55, wherein the matrices include model view matrices.

57. The system as recited in claim 56, wherein the additional processing includes multiplying the sum of products by a composite matrix for displaying purposes.

58. The system as recited in claim 56, wherein the additional processing includes a lighting operation.

59. The system as recited in claim 55, wherein the matrices include inverse matrices and the vertex data includes a normal vector.

60. The system as recited in claim 55, wherein the single integrated circuit includes: a multiplication logic unit having a first input coupled to an output of the buffer for receiving the vertex data; an arithmetic logic unit having a first input coupled to an output of the multiplication logic unit; a register unit having an input coupled to an output of the arithmetic logic unit; memory coupled to the multiplication logic unit and the arithmetic logic unit, the memory having stored therein a plurality of constants and variables for being used when processing the vertex data.

61. The system as recited in claim 55, wherein the single integrated circuit includes: a multiplication logic unit having a first input coupled to an output of the buffer; an arithmetic logic unit having a first input coupled to an output of the multiplication logic unit; a register unit having an input coupled to an output of the arithmetic logic unit; an inverse logic unit including an input coupled to the output of the arithmetic logic unit or the register unit for

performing an inverse or an inverse square root operation; a conversion module coupled between an output of the inverse logic unit and a second input of the multiplication logic unit, the conversion module adapted to convert scalar vertex data to vector vertex data; memory coupled to the multiplication logic unit and the arithmetic logic unit, the memory having stored therein a plurality of constants and variables for being used when processing the vertex data; and an output converter coupled to the output of the arithmetic logic unit for being coupled to a lighting module to output the processed vertex data thereto.

62. A method for handling output values in a graphics processing module representative of an inverse operation involving a W-attribute of vertex data, comprising:

(a) processing vertex data, wherein the processing of the vertex data includes an inverse operation involving a W-attribute of the vertex data;

(b) outputting the processed vertex data;

(c) identifying a value of the inverse operation involving the W-attribute of the vertex data; and

(d) clamping the value of the inverse operation if the value of the inverse operation meets predetermined criteria.

63. The method as recited in claim 62, wherein the criteria includes the value of the inverse operation being greater than a predetermined amount.

64. The method as recited in claim 62, wherein the value is clamped by an inverse logic unit in a transform module.

65. The method as recited in claim 62, wherein the value is clamped to a minimum and a maximum exponent.

66. A computer program embodied on a computer readable medium for handling output values in a graphics processing module representative of an inverse operation involving a W-attribute of vertex data, comprising:

(a) a code segment for processing vertex data, wherein the processing of the vertex data includes an inverse operation involving a W-attribute of the vertex data;

(b) a code segment for outputting the processed vertex data;

(c) a code segment for identifying a value of the inverse operation involving the W-attribute of the vertex data; and

(d) a code segment for clamping the value of the inverse operation if the value of the inverse operation meets predetermined criteria.

67. The computer program as recited in claim 66, wherein the criteria includes the value of the inverse operation being greater than a predetermined amount.

68. The computer program as recited in claim 66, wherein the value is clamped by an inverse logic unit in a transform module.

69. The computer program as recited in claim 66, wherein the value is clamped to a minimum and a maximum exponent.